



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/008,952

12/06/2001

Ashley K. Wise

RA5417 (USYS030.PA)

3730

27516

7590

04/24/2006

UNISYS CORPORATION

MS 4773

PO BOX 64942

ST. PAUL, MN 55164-0942

EXAMINER

MITCHELL, JASON D

ART UNIT

PAPER NUMBER

2193

DATE MAILED: 04/24/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	Application No. 10/008,952	Applicant(s) WISE, ASHLEY K.	
	Examiner Jason Mitchell	Art Unit 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) ☒ Responsive to communication(s) filed on 10 January 2006.
- 2a) ☒ This action is **FINAL**.                      2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) ☒ Claim(s) 1-4 and 6-20 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-4 and 6-20 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All    b) ☐ Some \*    c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)  | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                                   | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)             |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____  |

### **DETAILED ACTION**

This action is in response to remarks filed on 1/10/06.

No amendments have been made to the currently pending claims. Claims 1-4 and 6-20 are now pending in this case.

### ***Response to Arguments***

Examiner notes that the Carey reference was omitted in the rejection headings of the previous action. This oversight is corrected in this action.

**Applicant's arguments have been fully considered but they are not persuasive.**

Starting in the third paragraph on pg. 8 Applicant states:

The cited teachings of Carey neither teach nor suggest the claim limitations. The Office Action explains that "the teaching in Carey that is relied upon is simply the application of an optimal range of available nodes in a free list (minimum and maximum thresholds)." Looking then to how Carey accomplishes this, it can be seen that the approach taught by Carey does not reasonably correspond to or even suggest the claim limitations.

... Carey, in order to achieve the desired number of available free pages, does not allocate and deallocate memory as claimed. Rather, Carey teaches returning pages that are used to a list of free page buffers.

... Carey reclaims page buffers that are in use, and Carey's approach would render the White-Hardy combination inoperable.

Respectfully, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would

Art Unit: 2193

have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

It is acknowledged that 'Carey reclaims page buffers that are in use' and implementing this type of allocation / deallocation would 'render the White-Hardy combination inoperable'. However it is Examiner's position that one of ordinary skill in the art would have recognized this, and accordingly would not have replaced the existing allocation / deallocation means taught in Hardy (col. 8, lines 4-27) with those of Carey, thus creating a free list manager which uses conventional allocation / deallocation from / to memory (col. 8, lines 8-10 'memory will be allocated) to maintain a number of free nodes between lower and upper thresholds (col. 7, lines 44-46 'minimum threshold ... maximum threshold') as is required by the claims.

In the second to last full paragraph on pt. 9 Applicant states:

The Office Action further fails to show that the White-Hardy combination suggests the limitation of storing a numerical value in the allocated plurality of storage nodes and forming a linked list of the allocated plurality of storage nodes.

The Office Action acknowledges that White "does not detail methods for memory management," and goes on to allege that "one of ordinary skill in the art would have to look to the prior art (i.e. Hardy) to provide such an implementation." However, there is no evidence presented that supports the assertion that one would "have"; to look to Hardy's prior art teachings to implement memory management for White's bignums.

As stated previously, White is silent regarding the implementation of a memory management system. Thus one of ordinary skill in the art would be motivated to look to the prior art for some teaching regarding the implementation of a memory management system in order to implement a system to manage memory. Examiner did not intend to

Art Unit: 2193

argue that one would "have" to look to Hardy specifically, but that Hardy is one of a plurality of equally obvious references, which could be used to provide the implementation details of a memory management system, which were omitted by White. Further, Hardy provides several aspects that would be considered beneficial by one of ordinary skill in the art. For instance, allowing pointer values to be 'made as small as possible yet still [able to] access the maximum amount of memory available' (col. 8, lines), eliminating 'the need for rearrangement of allocated memory into contiguous blocks ("garbage collection") or consolidation of memory' (col. 8, lines 14-18), and quick and easy access to the memory (col. 8, lines 18-21). These aspects of the Hardy system provide efficient memory management (Hardy col. 8, line 4 'memory must be managed efficiently'), and one of ordinary skill in the art would have found it useful to include these methods in Whites system (pg. 176, col. 2, par. 3).

Starting in the first full paragraph on pg. 10 Applicant states:

It is well-established that clear and particular evidence of making the combination must be provided in order to establish a prima facie case of obviousness.

...

The alleged motivation draws the conclusion, without supporting evidence, that White's memory management is less efficient than that taught by Hardy. Therefore, the alleged motivation to combine Hardy with White is insufficient to support prima facie obviousness.

In *Ruiz v. A.B. Chance Co.*, 69 USPQ 2d 1686 (CA FC 2004) the court found

While this court indeed warns against employing hindsight, its counsel is just that – a warning. That warning does not provide a rule of law that an express, written motivation to combine must appear in prior art references before a finding of obviousness. Stated differently, this court has consistently stated that a court or examiner may find a motivation to combine prior art references in the nature of the problem to be solved. See *Pro-Mold*, 75 F.3d at 1573; *Display Techs., Inc. v. Paul Flum*.

Art Unit: 2193

*Ideas, Inc.*, 282 F.3d 1340, 1346-47 (Fed. Cir. 2002); *In re Huang*, 100 F.3d 135, 139 n.5 [40 USPQ2d 1685] (Fed. Cir. 1996).

In light of the court's findings it can be seen that one of ordinary skill in the art would have found further motivation to combine the White and Hardy references, because both seek to solve the problem of memory management (White, pg. 176, col. 2, par. 3; Hardy, col. 8, lines 18-21 'A free list of available nodes').

In the second to last paragraph on pg. 10 Applicant states:

The alleged motivation draws the conclusion, without supporting evidence, that White's memory management is less efficient than that taught by Hardy. Therefore, the alleged motivation to combine Hardy with White is insufficient to support prima facie obviousness.

Examiner respectfully disagrees. Even assuming, for the sake of argument, that White's memory management is more efficient than Hardy's, prima facie obviousness is not ruled out.

"A known or obvious composition does not become patentable simply because it has been described as somewhat inferior to some other product for the same use." *In re Gurley*, 27 F.3d 551, 554, 31 USPQ2d 1130, 1132 (Fed. Cir. 1994)

In the paragraph bridging pp. 10 and 11, Applicant states:

The Office Action cites White's "seamless interface between fixnums and bignums as suggesting the limitations of overloading language-provided memory allocation and deallocation operations with large-integer operators that allocate and deallocate storage nodes. However, even if one assumes that White's numerical operators are overloaded, it does not necessarily follow that memory allocation operators would also have to be overloaded to support allocation and deallocation of storage nodes as claimed. Furthermore, no evidence is presented that indicates overloading memory operators necessarily follows from overloading numerical operators. Thus the Office Action fails to show that the limitations of claim 10 are suggested by White.

Providing a 'user invisible transition' between fixnums and bignums would certainly imply to one of ordinary skill in the art that **any action** on a fixnum use the same function calls and /or operators as would be used on a bignum, and vice-versa. If this were not the case then the distinction would be visible to the user, contradicting White's disclosure on pg. 174, col. 1, par. 3 - col. 2, par. 1 'a smooth, user invisible transition between ... fixnums—and those of larger size'.

In the first paragraph on pg. 11, Applicant states:

It is respectfully submitted that Knuth does not show recursion as understood by those skilled in the art. Knuth's algorithm D shows a program loop implementation, and program D on page 259 is assembly language in which there is not apparent recursion. Furthermore, two recursion steps are claimed for different parts of the dividend, where as Knuth shows a single loop. An explanation of that part of Knuth's algorithm understood to teach recursion is respectfully requested

Respectfully, Knuth's algorithm is recursive in that it repeatedly performs the same steps (D3-D6) on a smaller and smaller subset of the initial data (' $(u_j u_{j+1} \dots u_{j+n})_b$ ') until it reaches an ending or base condition ( $j > m$ , in other words no smaller subset exists).

Further Knuth clearly discloses multiple recursive calls on progressively less significant subsets of the dividend ('increase j by one'; 'a division of  $(u_j u_{j+1} \dots u_{j+n})_b$ ').

Further, it should come as no surprise to one of ordinary skill in the art that the algorithm would be implemented iteratively in a language like Assembly, which does not provide function calls and therefore does not support recursion.

Art Unit: 2193

In light of the above discussion, Examiner maintains the rejections of claims 1-4, 6-12, 14 and 17-20.

Starting in the last full paragraph on pg. 11 regarding claims 13, 15-16 and 19, Applicant argues that "The Office Action fails to show that all the limitations are suggested by the references and fails to provide a proper motivation for modifying the teachings of the White-Hardy combination with the teachings" of the respective references. But fails to provide any support for these statements, which therefore fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Accordingly the associated rejections of claims 12, 15-16 and 19 are maintained.

### ***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

**Claims 1-4, 6-12, 14, 17-18 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in**



**view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 6,078,994 to Carey (Carey).**

**Regarding Claims 1 and 18:** White discloses a computer-implemented method for processing numerical values in a computer program executable on a computer system, comprising: encapsulating in a large-integer datatype, large-integer data and associated operators (pg. 174, col. 1, par. 3 'indefinitely large integers—have been a part of Lisp for a long time'), wherein the large-integer data has runtime expandable precision (pg. 177, col. 2, par. 2 'allocated in units of at least one 32-bit word') and maximum precision is limited only by system memory availability (pg. 174, col. 1, par. 3 'indefinitely large integers'); and overloading language-provided arithmetic, logical, and type conversion operators with the large-integer operators that operate on large-integer variables in combination with other datatypes, and programmed usage of a variable of the large-integer datatype is equivalent to and interoperable with a variable of a system-defined integral datatype (pg. 174, col. 1, par. 3 - col. 2, par. 1 'a smooth, user invisible transition between ... fixnums—and those of larger size');

White does not disclose establishing a plurality of available storage nodes available for allocation to large-integer data; or allocating a subset of the plurality of available storage nodes for a large-integer variable, the subset being an allocated plurality of storage nodes, or storing a numerical value in the allocated plurality of storage nodes or forming a linked list of the allocated plurality of storage nodes. He does however disclose that Bignums are allocated memory in pre-sized chunks (pg. 176, col. 2, par. 3 'primitives ... to allocate memory for one of a given size').

Art Unit: 2193

Hardy discloses establishing a plurality of available storage nodes available for allocation (col. 8, lines 18-21 'A free list of available nodes'); and allocating a subset of the plurality of available storage nodes for a variable (col. 8, lines 8-10 'memory will be allocated ... for the pixel value nodes'), the subset being an allocated plurality of storage nodes, and forming a linked list of the allocated plurality of storage nodes (col. 8, lines 8-10 'nodes of the linked list) in an analogous art for the purpose of memory management (Hardy col. 8, line 4 'memory must be managed efficiently').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use Hardy's methods of memory allocation/de-allocation (col. 8, lines 4-27) with White's invention (pg. 176, col. 2, par. 3) to provide memory space for White's Bignums (pg. 177, par. 1 'Bignums are allocated in units of at least one 32-bit word') because one of ordinary skill in the art would have been motivated to provide an efficient memory management system (Hardy col. 8, line 4 'memory must be managed efficiently') to support White's disclosure of memory allocation (pg. 176, col. 2, par. 3). Further, with regard to claim 1, Hardy does not explicitly teach maintaining a minimum or maximum number of available storage nodes, but does teach maintaining a list of available storage nodes (col. 8, lines 18-21 'A free list of available nodes 39 is kept within each memory block').

Carey teaches determining a total number of available storage nodes available for allocation to large-integer data (col. 7, lines 40-42 'maintains a counter of the number of entries on the free list'); allocating memory for a first number of available storage nodes, responsive to the total number being less than first threshold value, and establishing the

first number of available storage nodes (col. 7, lines 44-46 'If the minimum threshold is met ... begins a collecting operation'); and halting collection of available storage nodes, responsive to the total number being greater than a second threshold value (col. 8, lines 44-46 'If this number is above a preset maximum threshold then processing [suspends]').

It would have been obvious to a person of ordinary skill in the art at the time of the invention track the number of nodes in Hardy's 'free list' (col. 8, lines 18-21 'A free list of available nodes') thereby maintaining at least a minimum number of nodes as taught in Carey (col. 7, lines 44-46 'the minimum threshold') in order to avoid processing delays (Carey col. 7, lines 40-41 'minimize processing delays ... when the free list becomes empty'). Further in Hardy's system, where nodes are explicitly returned to the 'free list' (col. 8, lines 22-23 'As nodes are removed from a linked list, they are returned to the free list') and not collected as in Carey (col. 7, lines 44-46 'begins a collecting operation'), it would have also been obvious to remove nodes from the 'free list' when the count exceeded the maximum threshold taught in Carey (col. 8, lines 44-46 'If this number is above a preset maximum threshold) in order to maintain an optimal number of nodes in the free list (Carey col. 8, lines 49-51 'to optimize the cash memory').

**Regarding Claim 2:** The rejection of claim 1 is incorporated; further White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

"Common Lisp The Language, 2<sup>nd</sup> Edition" by Guy Steel et al. (Common Lisp) teaches converting a character string into large-integer data in response to a constant definition

Art Unit: 2193

statement (sec. 5.1.1, par 1 'all numbers ... are self-evaluation forms. When such an object is evaluated, that object ... is returned as the value of the form').

Therefore, through his use of COMMON LISP, White inherently discloses converting a character string into large-integer data in response to a constant definition statement.

**Regarding Claim 3:** The rejection of claim 2 is incorporated; further White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches converting large-integer data to and from a character string for input (sec. 22.1.1 par. 16 'After the entire token is read in, it will be interpreted either as ... or number'), output (sec. 22.1.6, par. 2-3 'How an expression is printed depends on its data type'), and serialization (sec. 22.1.1 par. 16 'it begins an extended token. After the entire token is read in').

Therefore, through his use of COMMON LISP, White inherently discloses converting large-integer data to and from a character string for input, output, and serialization.

**Regarding Claim 4:** The rejection of claim 1 is incorporated; further, White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches converting input data from language-provided input functions to large-integer data (sec. 22.1.1 par. 15 'After the entire token is read in, it will be interpreted either as ... or number'); and converting large-integer data to a format compatible with language-provided output functions (sec. 22.1.6, par. 2-3 'How an expression is printed depends on its data type').

Therefore, through his use of COMMON LISP, White inherently discloses converting input data from language-provided input functions to large-integer data and converting large-integer data to a format compatible with language-provided output functions.

**Regarding Claim 6:** The rejection of claim 1 is incorporated; further, White discloses allocating a selected number of bits for each storage node in response to a program-specified parameter (pg. 178, col. 1, par. 3 'the size of a bigit, ... will vary from implementation to implementation').

**Regarding Claim 7:** The rejection of claim 1 is incorporated; further, White discloses dynamically allocating a number of storage nodes for storage of the numerical value as a function of a size of the numerical value (pg. 177, col. 2, par 2 'Bignums are allocated in units of at least one 32-bit word').

**Regarding Claim 8:** The rejection of claim 7 is incorporated; further, White discloses storing in each node that is allocated to a large-integer variable, a subset of bit values that represent a numerical value (pg. 178, col. 1, par. 3 'a 'bigit' is a 'bignum digit' and is thus an integer between 0 and R-1 for some positive radix R').

**Regarding Claim 9:** The rejection of claim 8 is incorporated; further White explicitly discloses allocating a storage node to a large-integer variable while performing a large-integer operation that generates a numerical value and stores the numerical value in the variable (pg. 180, col. 1 'allocating memory space for the result of a bignum-by-bignum multiplication'), if a number of bit values required to represent the numerical value exceeds storage available in storage nodes allocated to the large-integer variable pg. 180, col. 1 'We don't want the multiplication routine to ... be caught short by one bit'),

and therefore inherently discloses maintaining a set of available storage nodes that are not allocated to any large-integer variable from which to allocate storage nodes.

However White does not explicitly disclose returning to the set of available storage nodes a storage node allocated to a large-integer variable while performing a large-integer operation that generates a numerical value for storage in the variable, if a number of bit values required to represent the numerical value is less than storage available in storage nodes allocated to the variable, but does disclose that 'We don't want ... to allocate extra space needlessly' (pg. 180, col. 1) and "The Art of Computer Programming" by Knuth, from which White derives his division algorithm (pg. 180, col. 2 'The division algorithm is essentially 'Algorithm D''), teaches, 'division of an  $(m + n)$ -place integer by an  $n$ -place integer, giving an  $(m + 1)$ -place quotient and an  $n$ -place remainder' (pg. 250, section 4.3.1).

Accordingly, It would have been obvious to a person of ordinary skill in the art at the time of the invention to return any 'extra' nodes to the pool of available storage nodes, when it is found that the number of nodes required to represent the numerical value (pg. 250, section 4.3.1 ' $m + 1$ ') is less than the number actually allocated (pg. 250, section 4.3.1 ' $m + n$ ') because one of ordinary skill in the art would have been motivated to collect the empty nodes (pg. 180, col. 1 'We don't want ... to allocate extra space needlessly').

**Regarding Claim 10:** The rejection of claim 9 is incorporated; further White discloses 'a smooth, user invisible transition between ... fixnums—and those of larger size' (pg. 174, col. 1, par. 3-col. 2, par. 1) and COMMON LISP similarly discloses a seamless interface

between fixnums and bignums (sec 2.1.1 par. 'Common Lisp is designed to hide this distinction').

One of ordinary skill in the art would understand that 'hiding the distinction' between fixnums and bignums would, at least, suggest a system in which all operations and functions can accept, without distinction, fixnums or bignums. One of ordinary skill in the art would also understand that a function, which can accept arguments of various types, as described above, is 'overloaded'.

Accordingly It would have been obvious to a person of ordinary skill in the art at the time of the invention to overload language-provided memory allocation and de-allocation operators with large-integer operators that allocate and de-allocate storage nodes

**Regarding Claim 11:** The rejection of claim 1 is incorporated; further White discloses the algorithm used for division is defined in "The Art of Computer Programming, Vol. II" by Knuth (Knuth) (pg. 177, col. 1, par. 2 'The particular algorithms used are ...described in section 4.3.1 of [Knuth 1981])

Knuth teaches identifying a set of most-significant bits of the dividend and a set of least-significant bits of the dividend (pg. 257, Algorithm D ' $v=(v_1v_2...v_n)_b$ '); recursively performing a large-integer divide operation using the set of most-significant bits as the input dividend (pg. 257, Algorithm D Step D2 'a division of  $(u_ju_{j+1}...u_{j+n})_b$ ', and returning a quotient and a remainder (Algorithm D step D4 'replace  $(u_ju_{j+1}...u_{j+n})_b$  by  $(u_ju_{j+1}...u_{j+n})_b$  minus  $q$  times  $(v_1v_2...v_n)_b$ '); finding a lower-part dividend as a function of the remainder and the set of least-significant bits (Algorithm D step D7 'increase  $j$  by one'); recursively performing a large-integer divide operation using the lower-part dividend (Algorithm D

Art Unit: 2193

steps D7 'go back to D3'); and concurrently solving for the quotient and the remainder (Algorithm D step D4 'minus q times  $(v_1v_2...v_n)_b$ ').

Therefore, through the use of the algorithm taught by Knuth, White implicitly discloses the limitations recited.

**Regarding Claim 12:** The rejection of claim 11 is incorporated; further White discloses identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function of a number of bits that represent the dividend and a number of bits that represent the divisor (pg. 178, col. 1, par. 3 'the size of a bigit ... will vary from ... algorithm to algorithm').

**Regarding Claim 14:** The rejection of claim 1 is incorporated; further White discloses emulating fixed-bit arithmetic on variables of the large-integer data type (pg. 174, col. 2, par. 4 'a set of new primitive arithmetic operations that focus on ... bignums').

**Regarding Claims 17 and 20:** The rejections of claims 1 and 18 are incorporated; further White does not explicitly disclose a large-rational datatype, but does disclose that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches a rational datatype (sec. 2.1.2, par. 1 'Integers and ratios collectively constitute the type rational') where a ratio is the mathematical ratio of two integers (sec. 2.1.2, par. 1), and integers encompass bignums (sec. 2.1.1, par. 2 'an integer that is not a fixnum is called a bignum'), thereby teaching the limitations recited in the instant claim as noted in the rejections of claims 1 and 18.



Therefore, through his use of COMMON LISP, White inherently discloses a large-rational datatype.

**Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 6,078,994 to Carey (Carey) further in view of "Fast Recursive Division" by Burnikel et al. (Burnikel).**

**Regarding Claim 13:** The rejection of claim 12 is incorporated; further the White-Hardy-Carey combination does not disclose identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function of one-half a difference between the number of bits that represent the dividend and the number of bits that represent the divisor. However, White does disclose the possibility of using various algorithms (pg. 177, col. 1, par. 2 'we have investigated some more complex algorithms').

Burnikel teaches a division algorithm identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function one-half a difference between the number of bits that represent the dividend and the number of bits that represent the divisor (pg. 4, par. 3 'dividing a  $2n$ -digit number by an  $n$ -digit number ... split  $A$  into four parts ... of length  $n/2$  each') in an analogous art for the purpose of improving the processing speed of a divide operation (pg. 1, par. 1 'our algorithm ... yields a speedup of more than 20%')

Art Unit: 2193

It would have been obvious to a person of ordinary skill in the art at the time of the invention to implement the division operation disclosed in White using the algorithm taught in Burnikel (pg. 4, Algorithm 1), because one of ordinary skill in the art would have been motivated to improve performance for a system where most of the division would be done for larger numbers (White pg. 177, col. 1, par. 2 'algorithms that do show a significant improvement in the asymptotic behaviors' and Burnikel pg. 4, par. 2 'Under the assumption that  $n$  is even and large').

**Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 6,078,994 to Carey (Carey) further in view of "Data Structures, an Advanced Approach Using C" by Esakov and Weiss (Esakov).**

**Regarding Claim 15:** The rejection of claim 1 is incorporated; further the White-Hardy-Carey combination does not disclose transferring data associated with temporary variables of the large-integer datatype by moving pointers to the data.

Esakov teaches transferring data by moving pointers to the data (pg. 16, par. 2, 'when an array is passed as an argument, a pointer is passed.') in an analogous art for the purpose of transferring data.

It would have been obvious to a person of ordinary skill in the art at the time of the invention to move a pointer to the data associated with temporary variables of the large-

integer datatype ('passed as an argument') in order to transfer the data more efficiently (Esakov pg. 16 'this saves the time and space of making a new copy').

**Claims 16 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 6,078,994 to Carey (Carey) further in view of US 5,619,711 to Anderson (Anderson).**

**Regarding Claims 16 and 19:** The rejections of claims 1 and 18 are incorporated; further the White-Hardy-Carey combination does not explicitly disclose a large-floating-point datatype, but does disclose that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Anderson teaches a large-floating-point datatype (col. 4, lines 15-17 'implementing infinite precision binary arithmetic') in an analogous art for the purpose of preserving numerical precision (col. 3, lines 15-17 'for preserving numerical precision').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use the teachings of Anderson to incorporate a large-floating-point datatype, using the techniques taught in Anderson (col. 4, lines 15-17), in a method similar to that disclosed for Bignums in White (pg. 174, col. 1, par. 3 'indefinitely large integers—have been a part of Lisp for a long time'), because one of ordinary skill in the art would have been motivated to 'provide special operation for floating point math'

Art Unit: 2193

(Anderson col. 2, lines 5-8), thereby teaching the limitations recited in the instant claim as noted in the rejections of claims 1 and 18.

### ***Conclusion***

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

**THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).


A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571) 272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.


Art Unit: 2193

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Jason Mitchell  
04/05/06



KAKALI CHAKI  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100